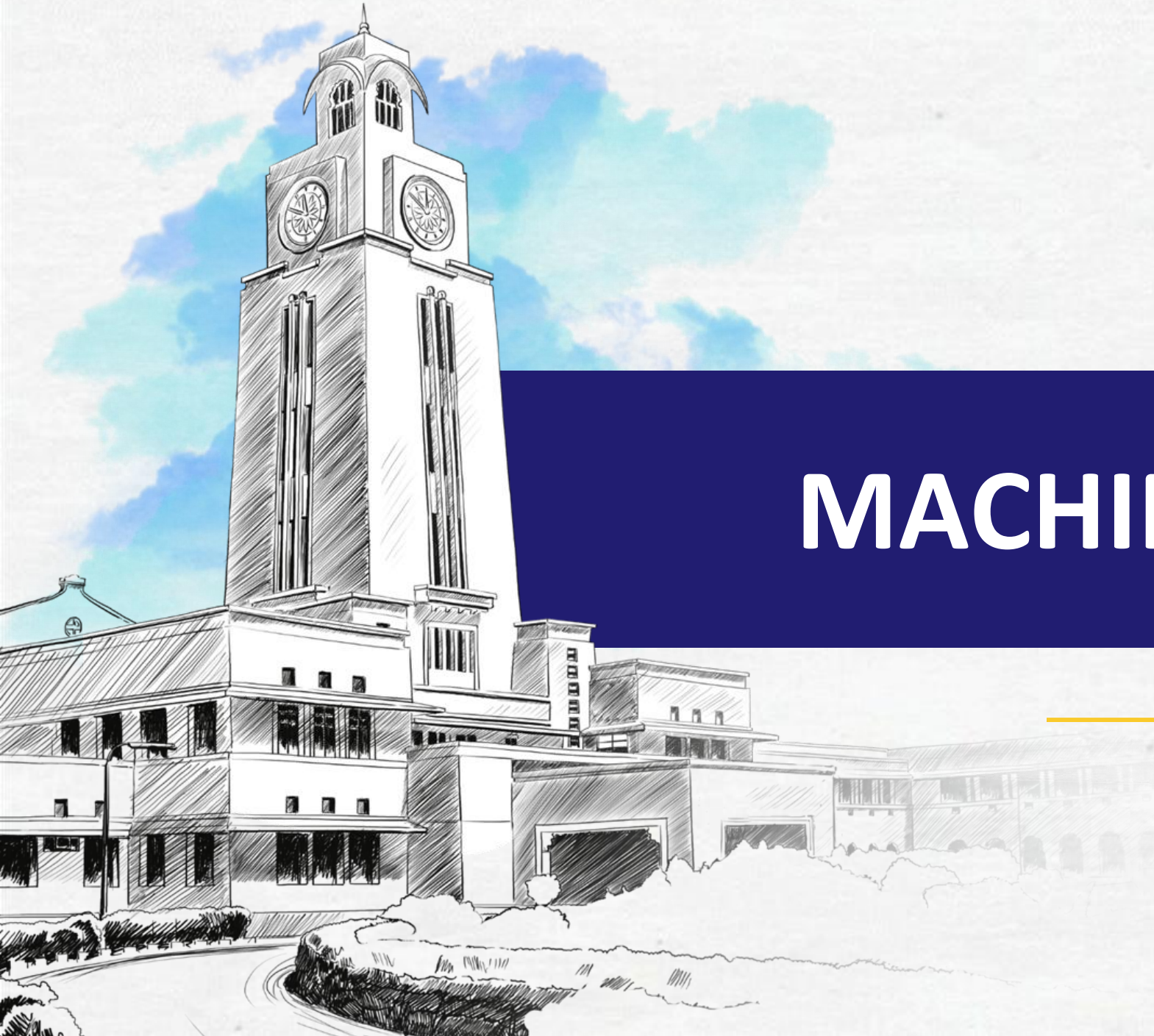


BITS Pilani
Pilani | Dubai | Goa | Hyderabad

MACHINE LEARNING

Session 5 & 6
(22nd February 2026)

Disclaimer and Acknowledgement

- The content for these slides has been obtained from books and various other source on the Internet
- I hereby acknowledge all the contributors for their material and inputs.
- I have provided source information wherever necessary
- I have added and modified the content flow to suit the requirements of the course and for ease of class presentation
- Students are requested to refer to the textbook and detailed content of this presentation deck over canvas

Recap

- Classification ✓
  - Output is discrete e.g. f(x) : {yes, no, maybe}
  - f(x) : {0, 1} or {yes, no} Binary Classification
  - Objective: minimize mis-classification rate

- Two kinds of Classifiers:
  - Probabilistic Generative Classifiers
    - Models aim to learn joint probability distribution P(X, Y)
  - Probabilistic Discriminative Classifiers
    - Models learn the decision boundary that separate the two classes

$P(X/Y)$

Eg: Naive Bayes', Gaussian Mixture Model, Linear Discriminant Analysis

it understands how the data is distributed within each class.

$P(Y|X)$

Eg: Logistic regression
Support Vector Machine (SVM)

Agenda

Logistic Regression

Course Plan

M1	Introduction
M2	Machine learning Workflow
M3	Linear Models for Regression
M4	Linear Models for Classification
M5	Decision Tree
M6	Instance Based Learning
M7	Support Vector Machine
M8	Bayesian Learning
M9	Ensemble Learning
M10	Unsupervised Learning
M11	Machine Learning Model Evaluation/Comparison

Logistic Regression

- statistical model used for binary classification problems,
  - goal is to predict the probability that a given input belongs to one of two classes.
  - uses the logistic (sigmoid) function to model the relationship between the input variables (also known as predictors or features) and the output variable (also known as the response or target)

    MODEL:

    z = woxo + w1 x1 + …. + wnxn

    hz(x) = 1/ 1 + e^ -z

$$z = x_0 w_0 + w_1 x_1 + w_2 x_2 + \dots + w_n x_n$$

$$h_z(x) = \frac{1}{1 + e^{-z}}$$ → sigmoid function or logistic function

z = new value

Logistic Regression vs Least Squares Regression

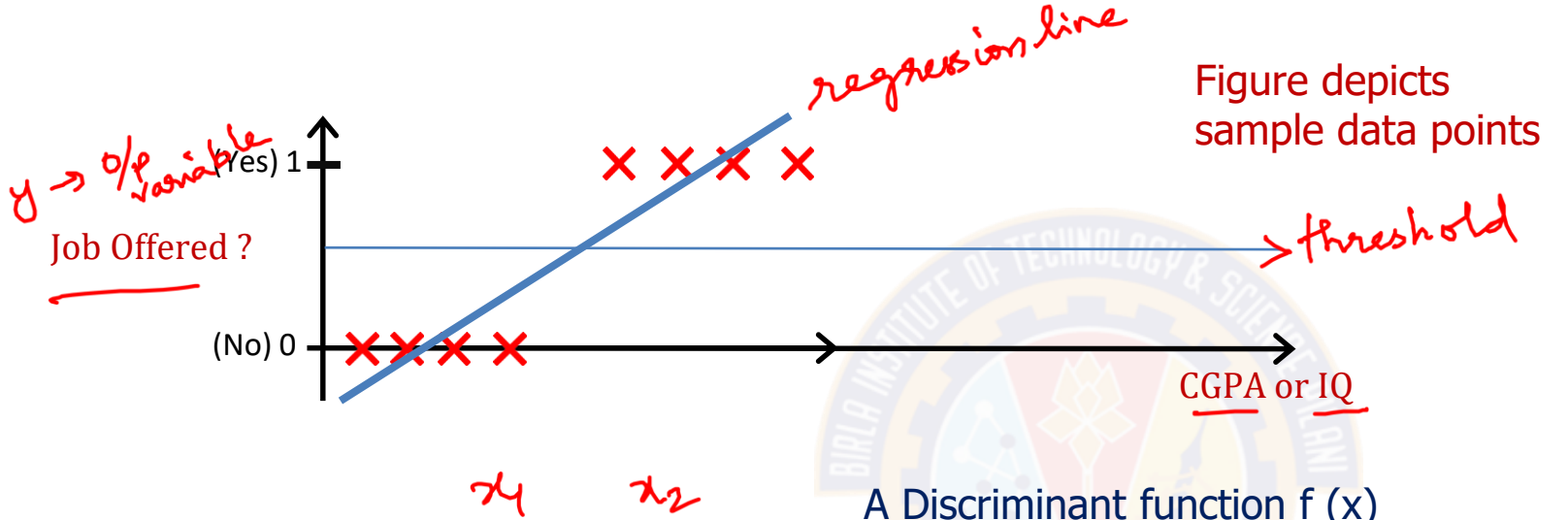


Figure depicts sample data points

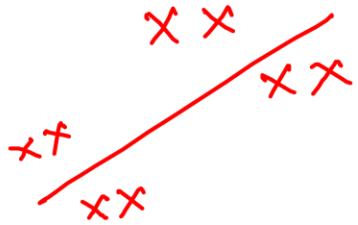
$y = h_\theta(x)$

- Independent Attribute : CGPA or IQ
- Can we solve the problem using linear regression? E.g., fit a straight line and define a threshold at 0.5
- Threshold classifier output $h_\theta(x)$ at 0.5:

A Discriminant function f (x) directly map input to class labels
In two-class problem, f (.) is binary valued

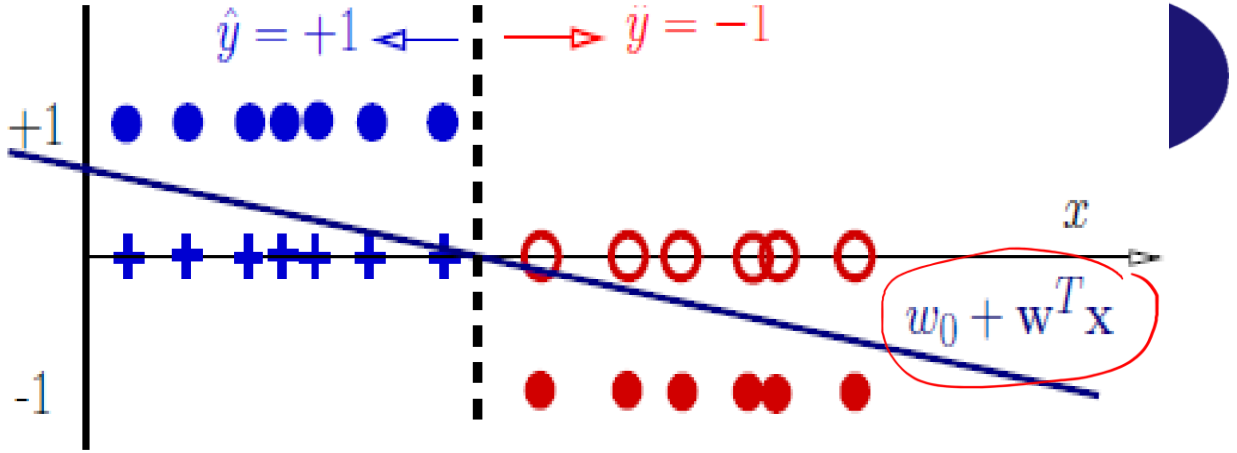
If $h_\theta(x) \geq 0.5$, predict "y = 1"

If $h_\theta(x) < 0.5$, predict "y = 0"



In this use case : $h_\theta(x) = 0.7$, implies that there is 70% of chance of the candidate being selected in the interview

Decision Rules



•Classifier:

$f(\mathbf{x}, \mathbf{w}) = w_o + \mathbf{w}^T\mathbf{x}$ (linear discriminant function)

•Decision rule is

$$y = \begin{cases} 1 & \text{if } f(\mathbf{x}, \mathbf{w}) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

•Mathematically

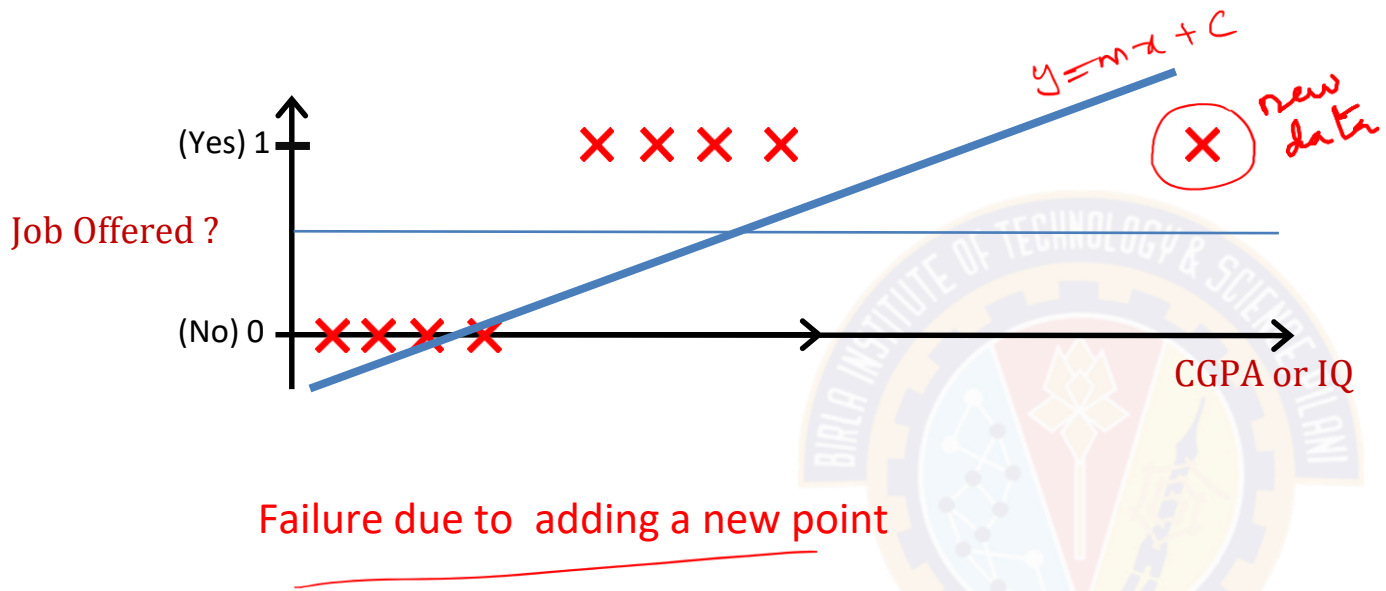
$y = \text{sign}(w_0 + \mathbf{w}^T\mathbf{x})$

•This specifies a linear classifier: it has a linear boundary (hyperplane)

$w_0 + \mathbf{w}^T\mathbf{x} = 0$

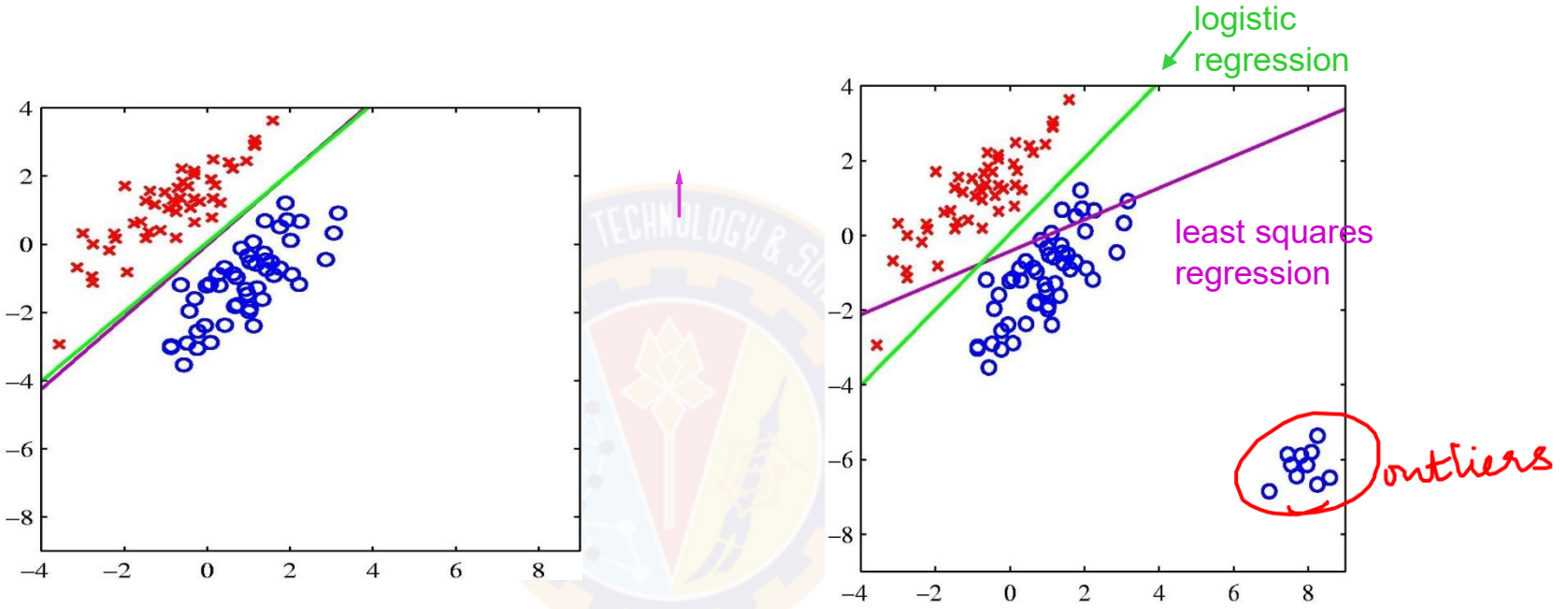
A discriminant is a function that takes an input vector x and assigns it to one of K classes, denoted C_k.

Logistic Regression vs Least Squares Regression



Failure due to adding a new point

Logistic Regression vs Least Squares Regression



The right-hand plot shows the corresponding results obtained when extra data points are added at the bottom left of the diagram, showing that least squares is highly sensitive to outliers, unlike logistic regression.

Sigmoid Function

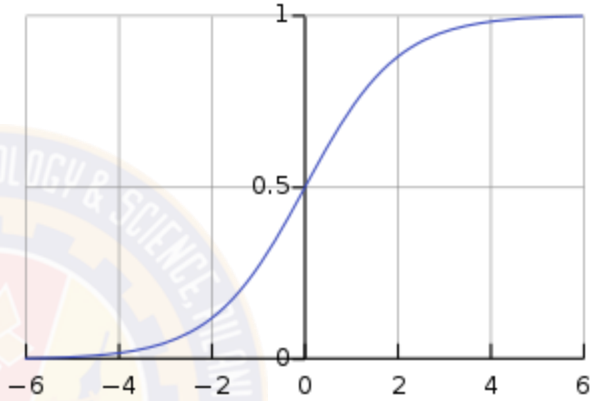
- Sigmoid/logistic function takes a real value as input and outputs another value between 0 and 1
- That framework is called logistic regression
  - Logistic: A special mathematical sigmoid function it uses
  - Regression: Combines a weight vector with observations to create an answer

- Want $0 \le h_\theta(x) \le 1$

- $h_\theta(x) = g(\theta^\top x)$,

where $g(z) = \frac{1}{1+e^{-z}}$

sigmoid function



$h_\theta(x) = g(\theta^T x)$

- Sigmoid function
- Logistic function

$$h_\theta(x) = \frac{1}{1+e^{-\theta^\top x}}$$

$\frac{1}{1+e^{-z}} = \frac{1}{1}$

$z=1$, $\frac{1}{1+e^{-1}} =$

Logistic Regression vs Least Squares Regression

Another drawback of using linear regression for this problem

$$Y = \theta_0 + \sum_i \theta_i x_i$$

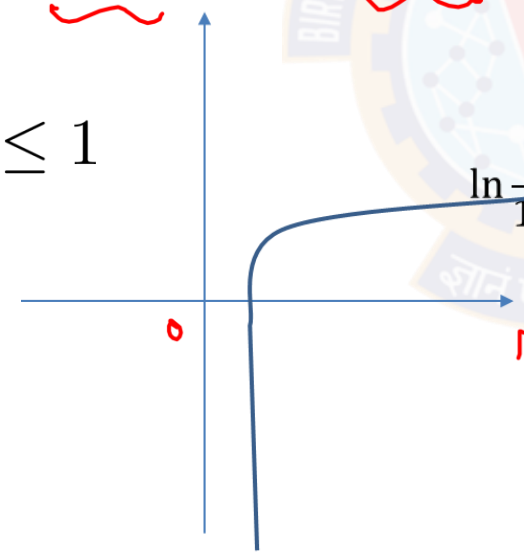
$$p = \frac{1}{1 + e^{-y}}$$

$h_\theta(x) = g(\theta^T x)$

sigmoid function

$$0 \leq h_\theta(x) \leq 1$$

$$\ln \frac{p}{1-p} = \theta_0 + \sum_i \theta_i x_i$$

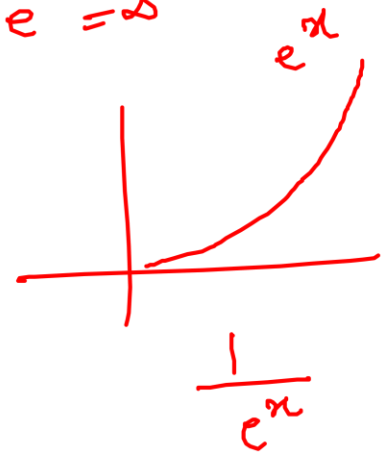


$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_i x_i$

$e^0 = 1$

$e^{-\infty} = 0$

$e^{\infty} = \infty$



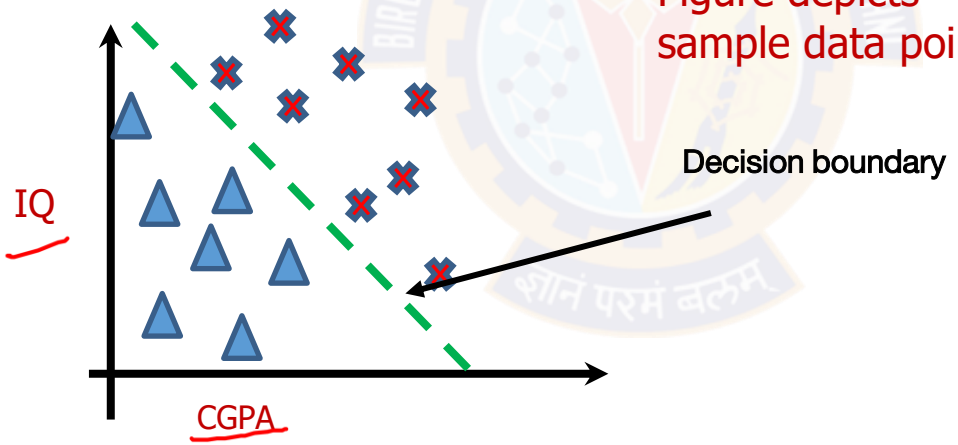
Logistic Regression – Sample Linear Boundary

$z = \theta_0 + \theta_1 x_1 + \theta_2 x_2$

- At decision boundary output of logistic regression is 0.5
- $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$
  - e.g., $\theta_0 = -3,\ \theta_1 = 1,\ \theta_2 = 1$

weights

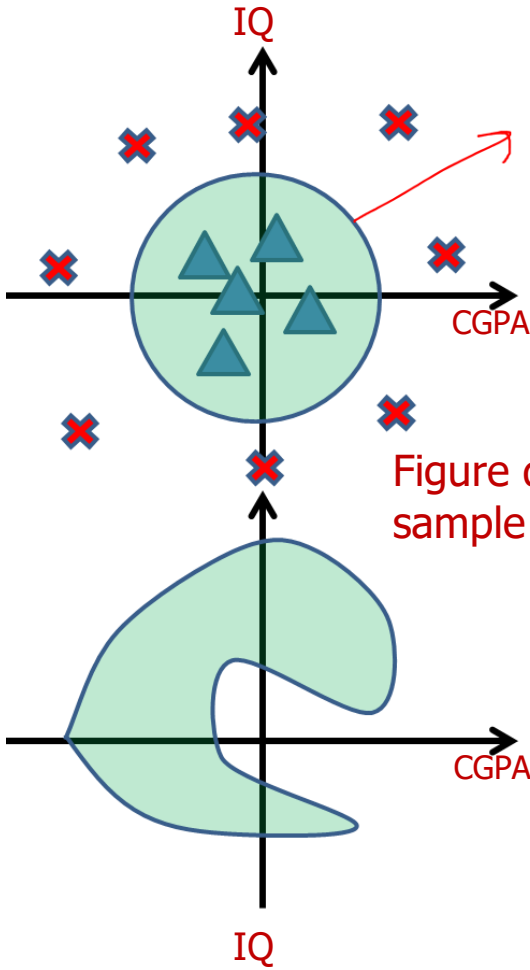
$h_z(x) = \frac{1}{1+e^{-z}}$



- Predict "$y = 1$" if $-3 + x_1 + x_2 \geq 0$

Slide credit: Andrew Ng

Logistic Regression – Sample Non-Linear



$x^2 + y^2 = 1$ unit circle

- $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$

E.g., $\theta_0 = -1, \theta_1 = 0, \theta_2 = 0, \theta_3 = 1, \theta_4 = 1$

- Predict "$y = 1$" if $-1 + x_1^2 + x_2^2 \geq 0$

$-1 + x_1^2 + x_2^2 \geq 0$

$x_1^2 + x_2^2 \geq 1$

Figure depicts sample data points

- $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^2 x_2^2 + \theta_6 x_1^3 x_2 + \cdots)$

Slide credit: Andrew Ng

Learning model parameters

- Training set:

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$$

- m examples

$$x \in \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix} \qquad x_0 = 1, y \in \{0, 1\}$$

bias

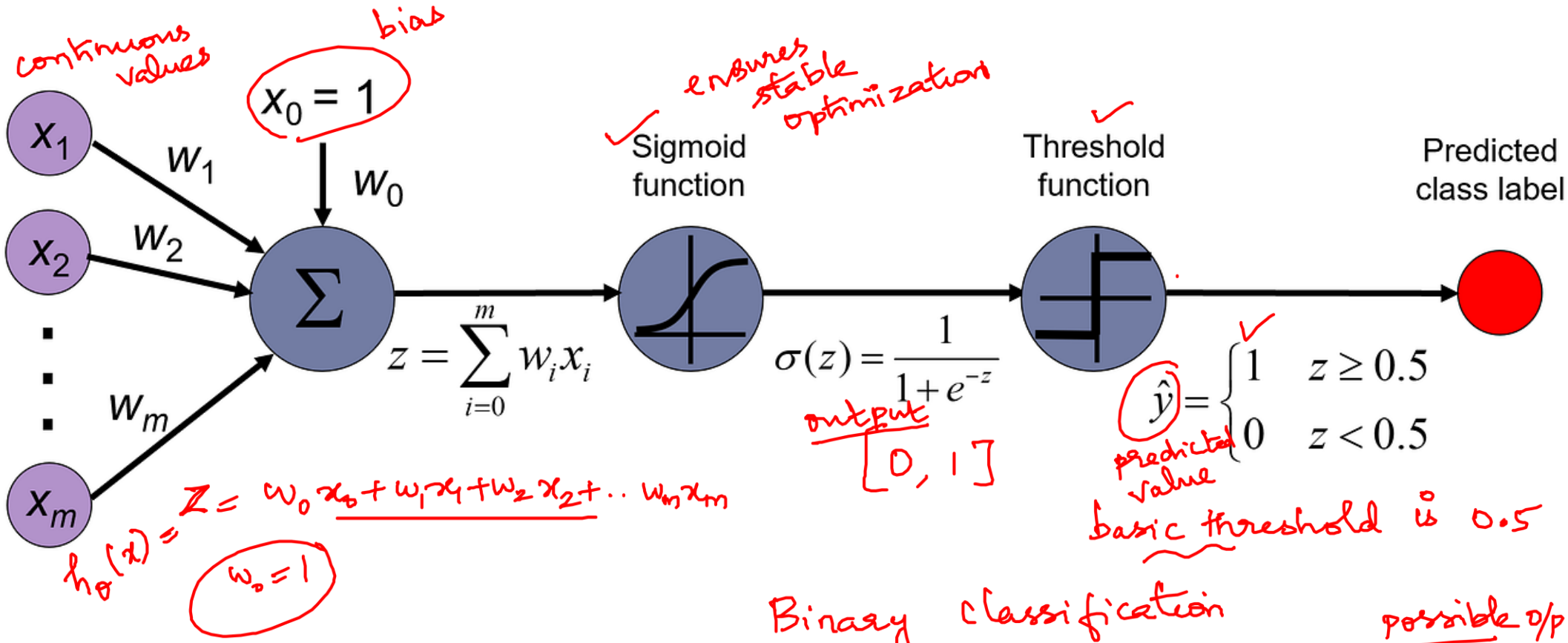
sigmoid fn.

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

- **How to choose parameters (feature weights) ?**

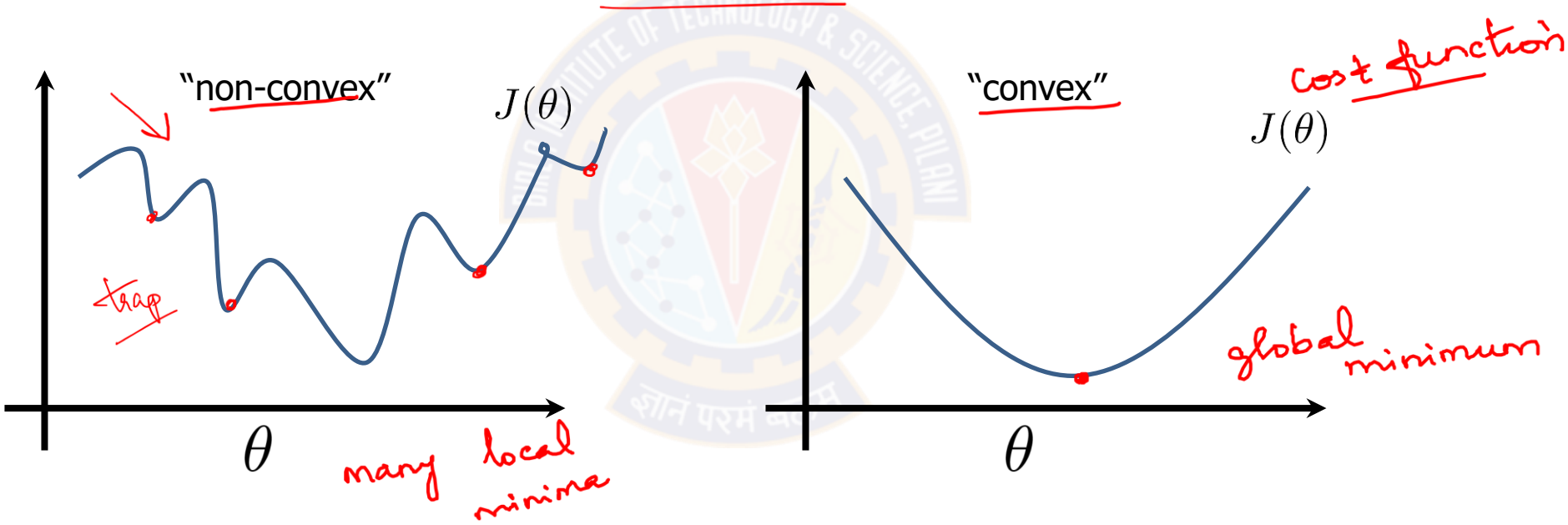
Notion of Cost Function in Classification

Logistic regression



Logistic Regression

- Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$
- How to choose parameters (feature weights)? θ



Logistic regression cost function (cross entropy)

MSE

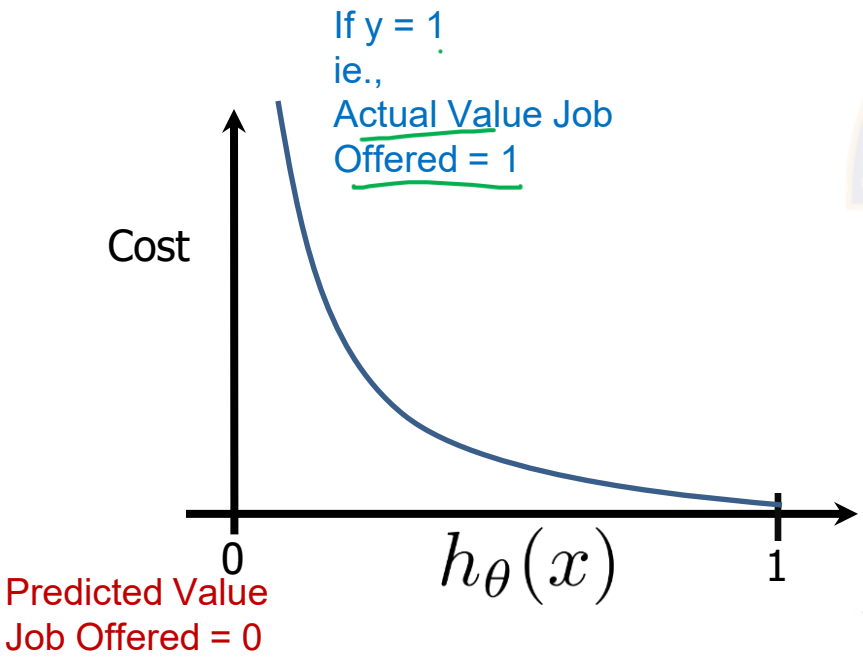
Log loss function

o/p of sigmoid fn.

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

difference b/w actual value y & predicted value $h_\theta(x)$

$0 \le h_\theta(x) \le 1$



$\text{Cost} = 0 \text{ if } y = 1, h_\theta(x) = 1$
But as $h_\theta(x) \to 0$
$Cost \to \infty$

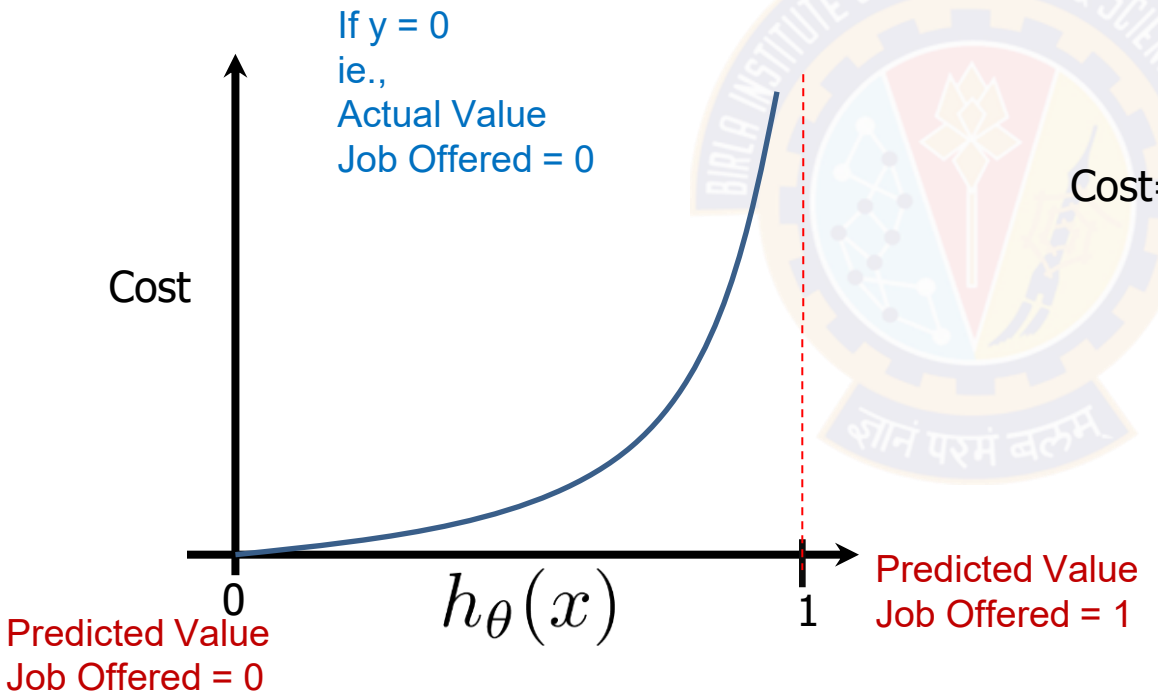
Captures intuition that if $h_\theta(x) = 0$, (predict $P(y = 1|x; \theta) = 0$), but $y = 1$, we'll penalize learning algorithm by a very large cost.

Predicted Value
Job Offered = 1

The minus sign is added because log probabilities are negative, and multiplying by −1 makes the cost positive and allows us to convert a maximum likelihood problem into a minimization problem.

Logistic regression cost function

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



Cost=0; If y=0 and $h_\theta(x)=0$

Cost function

Correct & final cost fn.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$= -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

How much error does the model make on average?"

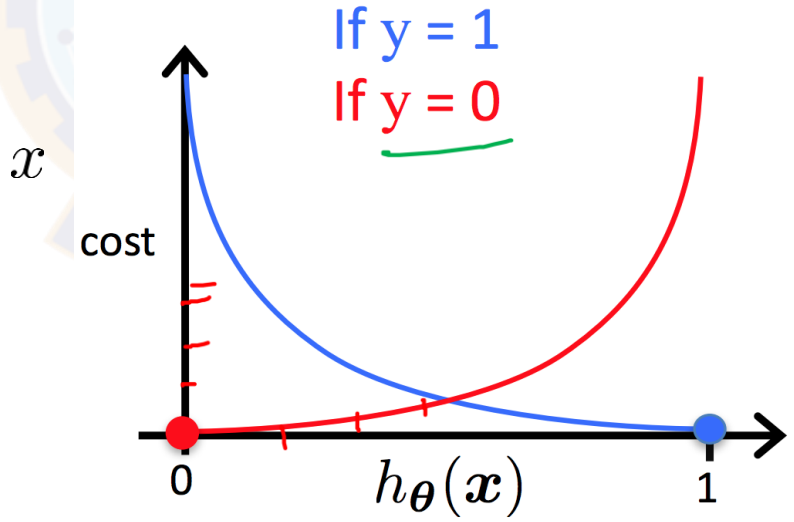
For eg:

$h_\theta(x) = 0.5 + 0.5\,\text{CGPA} + 0.5\,\text{IQ}$

To fit parameters θ : Apply Gradient Descent Algorithm $\min_\theta J(\theta)$

To make a prediction given new x :

Output : $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$



$h_\theta(x)$

carprice = 1.5 + Milage (0.5)

minimum error

carprice = 1 + 0.5 Milage + no. of years

Gradient Descent

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right) + (1 - y^{(i)}) \log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right]$$

Goal: $\min_\theta J(\theta)$

Repeat

{

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

}

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x_j^{(i)}$$

If we don't divide by m:

- Larger datasets → larger total cost
- Smaller datasets → smaller total cost

That would make comparison unfair.

- By dividing by m:
- Cost value stays in a comparable range
- It does not explode just because dataset is big

Gradient Descent Algorithm

- **Linear Regression**

Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right) x_j^{(i)}$$

}

$$h_\theta(x) = \theta^\top x$$

$= \theta_0 + \theta_1 x_1 + \theta_2 x_2$

Logistic Regression

Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right) x_j^{(i)}$$

}

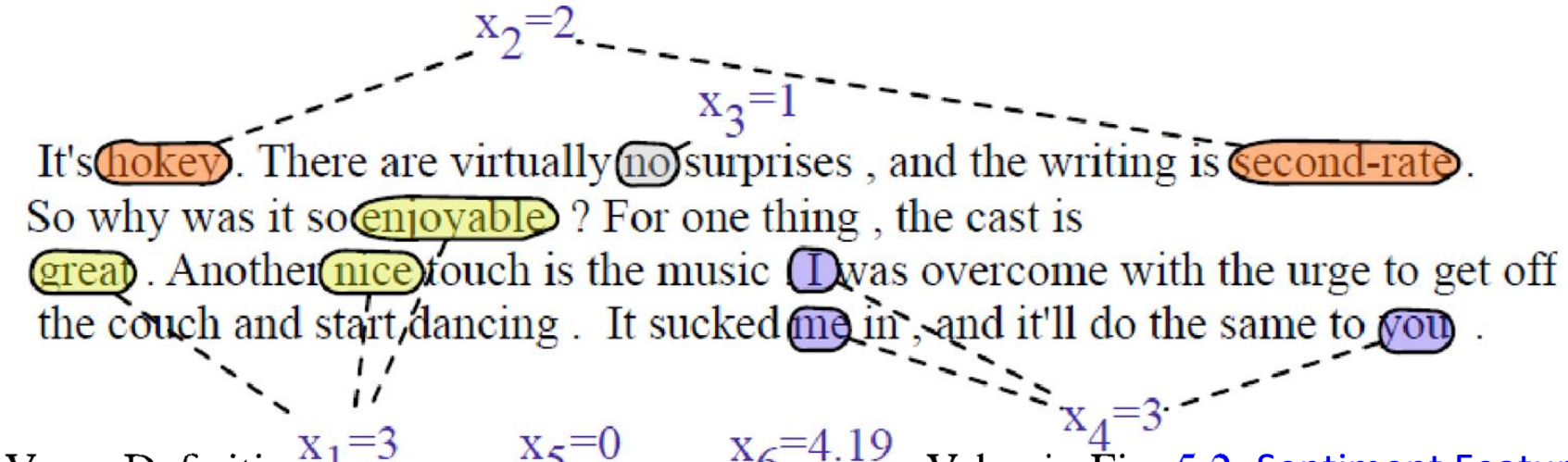
$$h_\theta(x) = \frac{1}{1 + e^{-\theta^\top x}}$$

$$h_\theta(x) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 CGPA + \theta_2 IQ)}}$$

Slide credit: Andrew Ng

Application of Logistic Regression
&
Problem Types

Example: Sentiment Analysis – With Engineered features


x2=2
x3=1
It's hokey . There are virtually no surprises , and the writing is second-rate .
So why was it so enjoyable ? For one thing , the cast is
great . Another nice touch is the music . I was overcome with the urge to get off
the couch and start dancing . It sucked me in , and it'll do the same to you .
x1=3
x5=0
x6=4.19
x4=3

Var	Definition	Value in Fig. 5.2
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	log(word count of doc)	$\ln(66) = 4.19$

Sentiment Features

$x = [3 \quad 2 \quad 1 \quad 3 \quad 0 \quad 4.19]$

Classifying sentiment using logistic

Answer

Suppose w = $[2.5, -5.0, -1.2, 0.5, 2.0, 0.7]$ given

$b = 0.1$

sigmoid function

$$p(+|x) = P(Y=1|x) = \sigma(w \cdot x + b)$$
$$= \sigma([2.5, -5.0, -1.2, 0.5, 2.0, 0.7] \cdot [3, 2, 1, 3, 0, 4.19] + 0.1)$$
$$= \sigma(.833)$$
$$= 0.70$$

$$p(-|x) = P(Y=0|x) = 1 - \sigma(w \cdot x + b)$$
$$= 0.30$$

Threshold = 0.5

BMI / DP Problem

$\theta_0 = 5$

$\theta_1 = -0.03$

$\theta_2 = -0.03$

$y = mx + b$

$= wx + b$ / $w_1 x_1 + x_0$

$w \cdot x = (2.5)(3) - 5(2) - 1.2(1) + 3(0.5) + 2(0) + 0.7(4.19)$

$h_\theta(x) = \sigma(x) = \frac{1}{1+e^{-x}} = \frac{1}{1+e^{-0.833}} = 0.70$

$= \sigma(0.833)$

Logistic Regression – Fit a Model – Apply Gradient

CGPA	IQ	IQ	Job Offered
5.5	6.7	100	1
5	7	105	0
8	6	90	1
9	7	105	1
6	8	120	0
7.5	7.3	110	0

Hyper parameters:
Learning Rate = 0.3
Initial Weights = (0.5, 0.5,0.5)
Regularization Constant = 0

$$\theta^T X = 0.5 + 0.5\ CGPA + 0.5\ IQ$$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

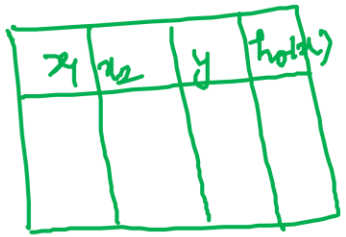
$$h_\theta(x) = \frac{1}{1 + e^{-(0.5 + 0.5\ CGPA + 0.5\ IQ)}}$$

Update the parameters

$$\theta_0 := \theta_0 - 0.3\frac{1}{6}\sum_{i=1}^{6}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) (1)$$

$$\theta_{CGPA} := \theta_{CGPA} - 0.3\frac{1}{6}\sum_{i=1}^{6}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x_{CGPA}^{(i)}$$

$$\theta_{IQ} := \theta_{IQ} - 0.3\frac{1}{6}\sum_{i=1}^{6}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x_{IQ}^{(i)}$$



Approx. : New weights

$\theta_0 = 0.4$

$\theta_{1=CGPA} = -0.4$

$\theta_{2=IQ} = -0.6$

Logistic Regression – Inference & Interpretation

CGPA	IQ	IQ	Job Offered
5.5	6.7	100	1
5	7	105	0
8	6	90	1
9	7	105	1
6	8	120	0
7.5	7.3	110	0

Assume : 0.4+0.3CGPA-0.45IQ

Predict the Job offered for a candidate : (5, 6)
h(x) = 0.31
Y-Predicted = 0 / No

Note :
The exponential function of the regression coefficient ($e^{w\text{-}cpga}$) is the odds ratio associated with a one-unit increase in the cgpa.
+ The odd of being offered with job increase by a factor of 1.35 for every unit increase in the CGPA
[np.exp(model.params)]

Regularization

Note : This topic is already covered in the module 3 and the implementation remains the same. **Only one points added here w.r.t interpretation for logistic regression**

Ways to Control Overfitting – Interpretation of Hyperparameter

- Regularization

$$Loss(S) = \sum_{i}^{n} Loss(\hat{y_i}, y_i) + \alpha \sum_{j}^{\#\,Weights} |\theta_j|$$

Note:
The hyperparameter controlling the regularization strength of a Scikit-Learn LogisticRegression model is not alpha (as in other linear models), but its inverse: C. The higher the value of C, the less the model is regularized.

Evaluation of Classifiers

Using Another Example

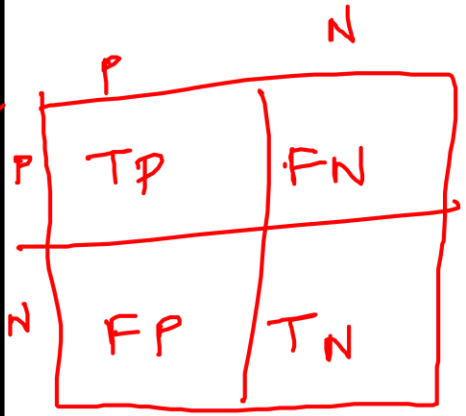
Confusion Matrix:

Given m classes, an entry, $CM_{i,j}$ in a **confusion matrix** indicates # of tuples in class i that were labeled by the classifier as class j

May have extra rows/columns to provide totals

- **True Positive (TP):** It refers to the number of predictions where the classifier correctly predicts the positive class as positive.
- **True Negative (TN):** It refers to the number of predictions where the classifier correctly predicts the negative class as negative.
- **False Positive (FP):** It refers to the number of predictions where the classifier incorrectly predicts the negative class as positive.
- **False Negative (FN):** It refers to the number of predictions where the classifier incorrectly predicts the positive class as negative.

Predicted class ->	C_1	$\neg C_1$
Actual class⇓		
C_1	**True Positives (TP)**	**False Negatives (FN)**
$\neg C_1$	**False Positives (FP)**	**True Negatives (TN)**



Evaluation of Classification Model

Mileage (in kmpl)	*Car Price (in cr)*
9.8	High
9.12	Low
9.5	High
10	Low
….	…

Confusion Matrix

		PREDICTED CLASS	
		Class= Low	Class= High
ACTUAL CLASS	Class= Low	a (TP)	b (FN)
	Class= High	c (FP)	d (TN)

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

Most effective when the class distribution is relatively balanced

Unseen Data	
Mileage (in kmpl)	***Car Price (in cr)***
7.5	High
10	Low
….	…..

Seller → FN ✓

Buyer → FN × (loss)

	High	Low
High	TP	FN
Low	FP	TN

TP → True Positive

FN → False Negative

FP → False Positive

TN → True Negative

Model 1

$$\text{CarPrice} = \frac{1}{1+e^{-8.5 + 0.5\ \text{Mileage} - 1.5\ \text{Mileage}^2}}$$

Accuracy: 99%

Model 2

$$\text{CarPrice} = \frac{1}{1+e^{5.5 - 1.5\ \text{Mileage}}}$$

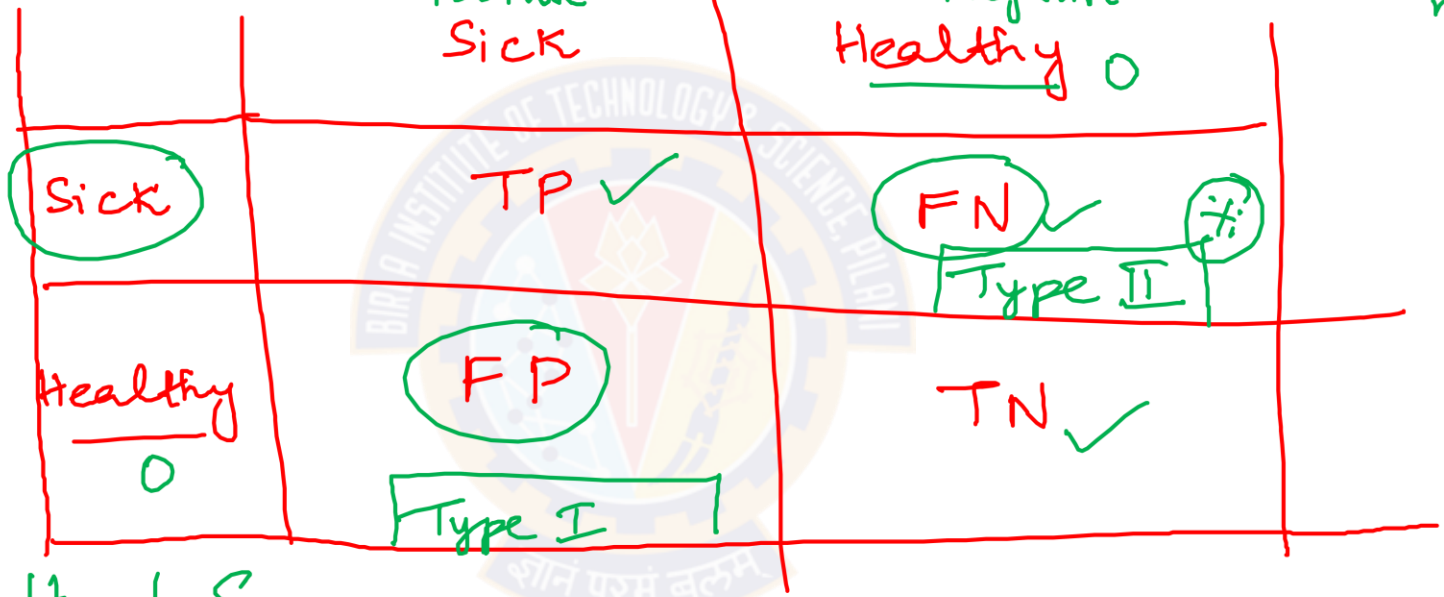
Accuracy: 50%

Evaluate Your Model

Test result

Type I → when a true H_0 is incorrectly rejected

Type II error: when a false H_0 is not rejected



P → class 1

N → class 0

+1 → Positive

-1 → Negative

	H	S
Healthy	TP	
Sick		TN

Healthy → class 0 → Negative Class

Sick → class 1 → Positive Class

Mileage	Actual car Price	Logistic Reg Predicted	O/P value
9.8	H	H	0.9
9.5	H	H	0.8
9.12	L	L	0.3
9.3	L	H	0.7

Threshold = 0.5

TP 2	FN 0
FP 1	TN 1

$$\text{Accuracy} = \frac{2+1}{2+1+1} = 3/4 = 75\%$$

$$\text{Car price} = \frac{1}{1+e^{-z}}$$

$$z = -8.5 + 0.5\,\text{Mil} - 1.5\,\text{Mil}^2$$

$$z_1 = -8.5 + 0.5(9.8) - 1.5(9.8)^2$$

$$\text{Predicted Value 1} = \frac{1}{1+e^{-z_1}}$$

$$\text{P. Value}_2 = \frac{1}{1+e^{-z_2}}$$

Evaluation of Classification Model

Confusion Matrix

		PREDICTED CLASS	
		Class= Low	Class= High
ACTUAL CLASS	Class= Low	0 (TP)	10 (FN)
	Class= High	0 (FP)	990 (TN)

If a model predicts everything to be class NO, accuracy is 990/1000 = 99 %. This is misleading because this trivial model does not detect any class YES example
Detecting the rare class is usually more interesting (e.g., frauds, intrusions, defects, etc)

		PREDICTED CLASS	
		Class= Low	Class= High
ACTUAL CLASS	Class= Low	10 (TP)	0 (FN)
	Class= High	500 (FP)	490 (TN)

imbalance

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

500/1000

Which model is better?

Model 1

$$\text{CarPrice} = \frac{1}{1+e^{-8.5 + 0.5\,\text{Mileage} - 1.5\,\text{Mileage}^2}}$$

Accuracy: 99%

Model 2

$$\text{CarPrice} = \frac{1}{1+e^{5.5-1.5\,\text{Mileage}}}$$

Accuracy: 50%

Evaluation of Classification Model

Confusion Matrix

		PREDICTED CLASS	
		Class= Low	Class= High
ACTUAL CLASS	Class= Low	a (TP)	b (FN)
	Class= High	c (FP)	d (TN)

The F-score (also known as the F1 score or F-measure, combines precision and recall into a single score . F1-score is a better metric when there are imbalanced classes (More on this in upcoming slides)

F-score
= 2 * (precision * recall) / (precision + recall)

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

$$ErrorRate = 1 - accuracy$$

$$Precision = Positive\ Predictive\ Value = \frac{TP}{TP + FP}$$

$$Recall = Sensitivity = TP\ Rate = \frac{TP}{TP + FN}$$

$$Specificity = TN\ Rate = \frac{TN}{TN + FP}$$

$$FP\ Rate = \alpha = \frac{FP}{TN + FP} = 1 - specificity$$

$$FN\ Rate = \beta = \frac{FN}{FN + TP} = 1 - sensitivity$$

$$Power = sensitivity = 1 - \beta$$

Definitions

- **Accuracy** is the proportion of all classifications that were correct, whether positive or negative.
- The **true positive rate (TPR)**, or the proportion of all actual positives that were classified correctly as positives, is also known as **recall**. Another name for recall is **probability of detection.** In an imbalanced dataset where the number of actual positives is very low, recall is a more meaningful metric than accuracy because it measures the ability of the model to correctly identify all positive instances.
- The **false positive rate (FPR)** is the proportion of all actual negatives that were classified *incorrectly* as positives, also known as the **probability of false alarm.**
- **Precision** is the proportion of all the model's positive classifications that are actually positive.

Which Classifier is better?

Case

Training set: Yes: 100, No: 100

T1	PREDICTED CLASS	
	Class=Yes	Class=No
ACTUAL CLASS — Class=Yes	50 (TP)	50 (FN)
ACTUAL CLASS — Class=No (Healthy)	1 (FP)	99 (TN)

Accuracy $\frac{50+99}{200} = 0.745$

T2	PREDICTED CLASS	
	Class=Yes	Class=No
ACTUAL CLASS — Class=Yes	99	1
ACTUAL CLASS — Class=No	10	90

$\frac{99+90}{200} = 0.945$

T3	PREDICTED CLASS	
	Class=Yes	Class=No
ACTUAL CLASS — Class=Yes	99	1
ACTUAL CLASS — Class=No	1	99

$\frac{99+99}{200} = 0.99$

Low Skew

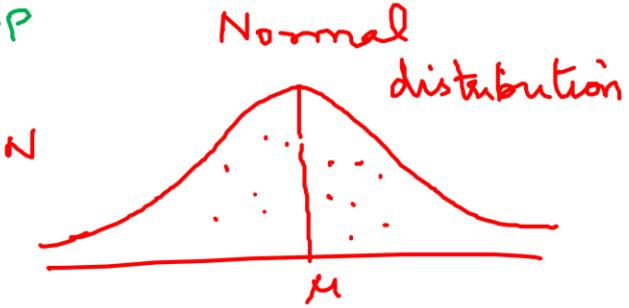
$$\text{Precision (p)} = \frac{TP}{TP+FP} = 0.98$$

$$\text{TPR} = \text{Recall (r)} = \frac{TP}{TP+FN} = 0.5$$

$$\text{FPR} = 0.01$$

$$\text{TPR/FPR} = 50$$

$$\text{F – measure} = 0.66$$



$$\text{Precision (p)} = 0.9$$

$$\text{TPR} = \text{Recall (r)} = 0.99$$

$$\text{FPR} = 0.1$$

$$\text{TPR/FPR} = 9.9$$

$$\text{F – measure} = 0.94$$

$$\text{Precision (p)} = 0.99$$

$$\text{TPR} = \text{Recall (r)} = 0.99$$

$$\text{FPR} = 0.01$$

$$\text{TPR/FPR} = 99$$

$$\text{F – measure} = 0.99$$

Precision → focuses on minimizing false positives

Which Classifier is better?

Medium Skew Case

T1	PREDICTED CLASS	
	Class=Yes	Class=No
ACTUAL CLASS — Class=Yes	50	50
ACTUAL CLASS — Class=No	10	990

$$\text{Precision (p)} = 0.83$$
$$\text{TPR} = \text{Recall (r)} = 0.5$$
$$\text{FPR} = 0.01$$
$$\text{TPR/FPR} = 50$$
$$\text{F} - \text{measure} = 0.62$$

T2	PREDICTED CLASS	
	Class=Yes	Class=No
ACTUAL CLASS — Class=Yes	99	1
ACTUAL CLASS — Class=No	100	900

$$\text{Precision (p)} = 0.5$$
$$\text{TPR} = \text{Recall (r)} = 0.99$$
$$\text{FPR} = 0.1$$
$$\text{TPR/FPR} = 9.9$$
$$\text{F} - \text{measure} = 0.66$$

T3	PREDICTED CLASS	
	Class=Yes	Class=No
ACTUAL CLASS — Class=Yes	99	1
ACTUAL CLASS — Class=No	10	990

More

$$\text{Precision (p)} = 0.9$$
$$\text{TPR} = \text{Recall (r)} = 0.99$$
$$\text{FPR} = 0.01$$
$$\text{TPR/FPR} = 99$$
$$\text{F} - \text{measure} = 0.94$$

Which Classifier is better?

High Skew Case

T1	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	50	50
	Class=No	100	9900

$$\text{Precision (p)} = 0.3$$
$$\text{TPR} = \text{Recall (r)} = 0.5$$
$$\text{FPR} = 0.01$$
$$\text{TPR/FPR} = 50$$
$$\text{F} - \text{measure} = 0.375$$

T2	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	99	1
	Class=No	1000	9000

$$\text{Precision (p)} = 0.09$$
$$\text{TPR} = \text{Recall (r)} = 0.99$$
$$\text{FPR} = 0.1$$
$$\text{TPR/FPR} = 9.9$$
$$\text{F} - \text{measure} = 0.165$$

T3	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	99	1
	Class=No	100	9900

Type I error

$$\text{Precision (p)} = 0.5$$
$$\text{TPR} = \text{Recall (r)} = 0.99$$
$$\text{FPR} = 0.01$$
$$\text{TPR/FPR} = 99$$
$$\text{F} - \text{measure} = 0.66$$

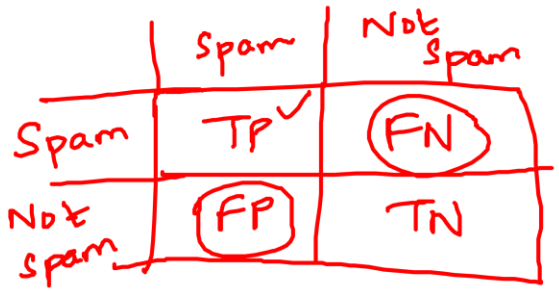
Which Model should you use?

	False Positive Rate	False Negative Rate
Model 1	41%	3%
Model 2	5%	25%

False Positive Rate (FPR): **how often negative samples are incorrectly classified as positive**.

Spam filtering

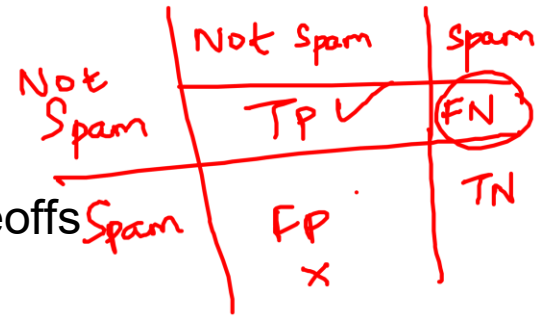
False Negative Rate (FNR): **how often positive samples are incorrectly classified as negative**.



Model 1
Model 2

Mistakes have different costs:

- Disease Screening – LOW FN Rate
- Spam filtering – LOW FP Rate

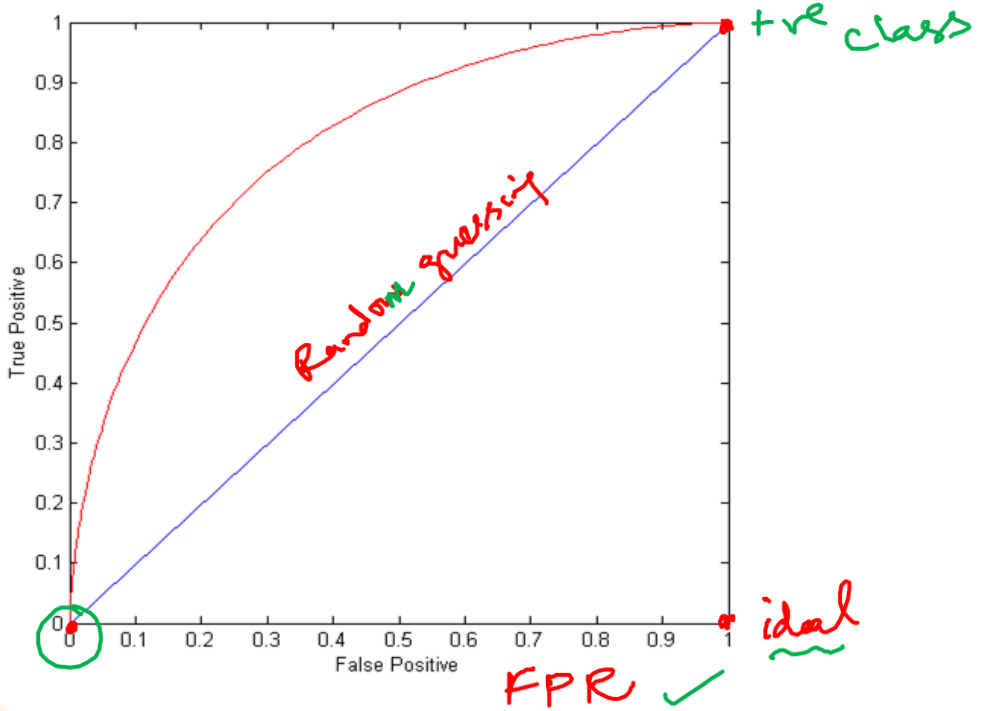


Conservative vs Aggressive settings:

- The same application might need multiple tradeoffs

ROC (Receiver Operating Characteristic)

- A graphical approach for displaying trade-off between detection rate and false alarm rate
- Developed in 1950s for signal detection theory to analyze noisy signals
- ROC curve plots TPR against FPR
  - Performance of a model represented as a point in an ROC curve



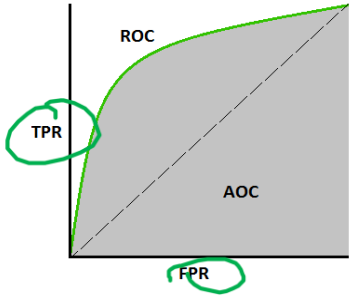
(TPR,FPR):

- (0,0): declare everything to be negative class
- (1,1): declare everything to be positive class
- (1,0): ideal
- Diagonal line:
  - Random guessing
  - Below diagonal line:
    - prediction is opposite of the true class

Receiver Operating Characteristic (ROC) Curve

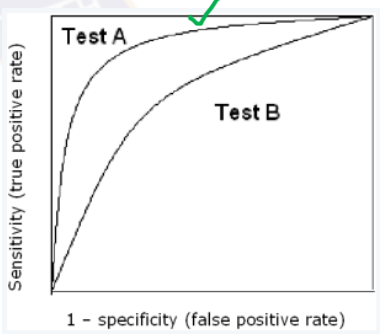
- ROC curve plots TPR and FPR, to graphically represent their trade-off
- AUC represents degree or measure of separability. It tells how much model is capable of distinguishing between classes.
  - Higher the AUC, better the model is at predicting
- Area Under Curve (AUC) of ROC – evaluates model performance on average
  - AUC of ROC = 1, for a perfect model
  - AUC of ROC = 0.5, if the model is random
- For model comparison, AUC of ROC should be larger for the model to be superior or better performing

AUC → Area Under the Curve

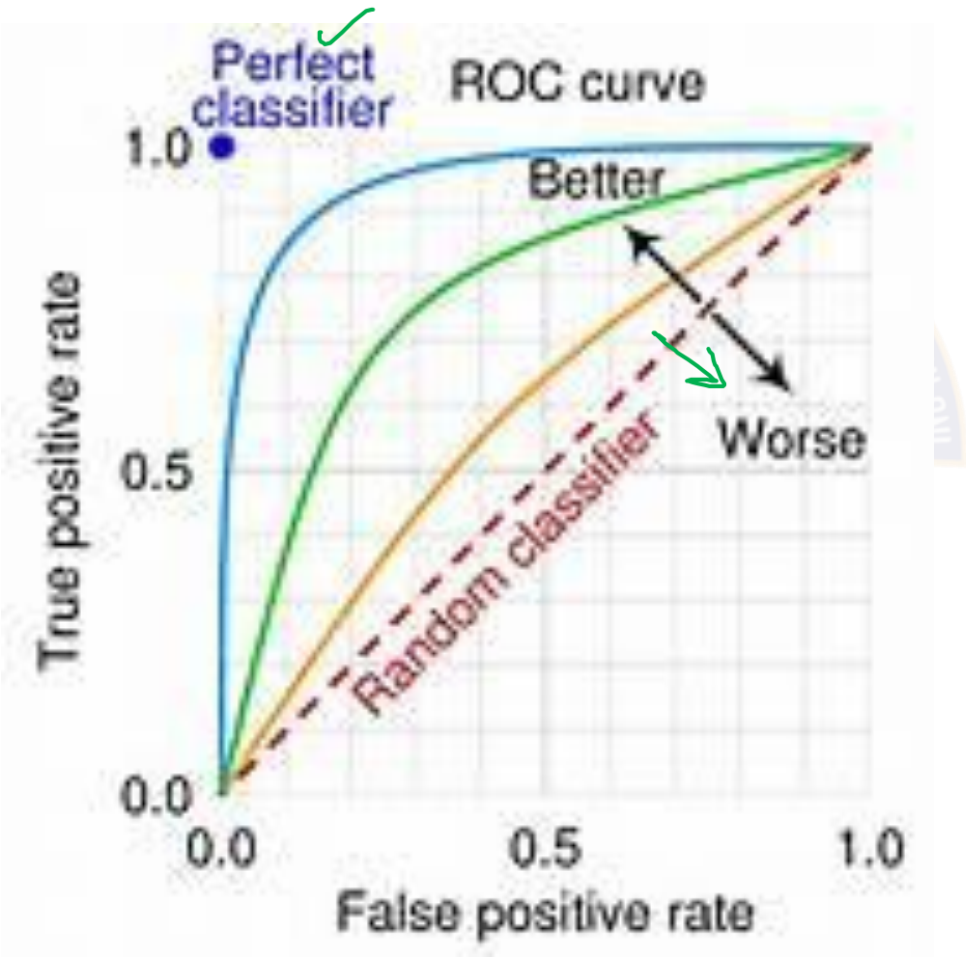


Usage

- Threshold selection
- Performance assessment
- Classifier comparison

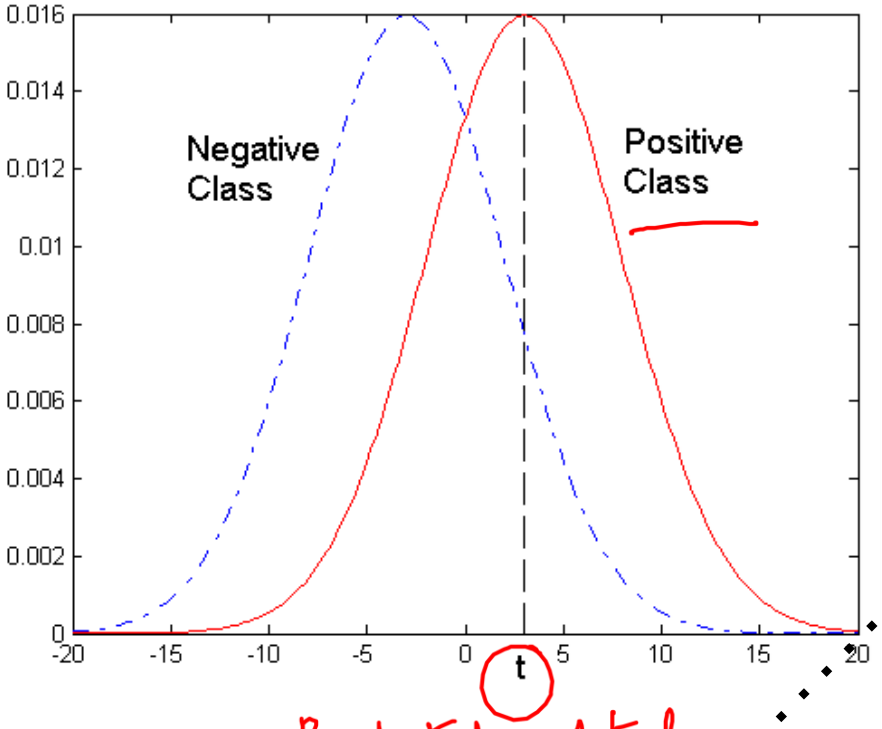


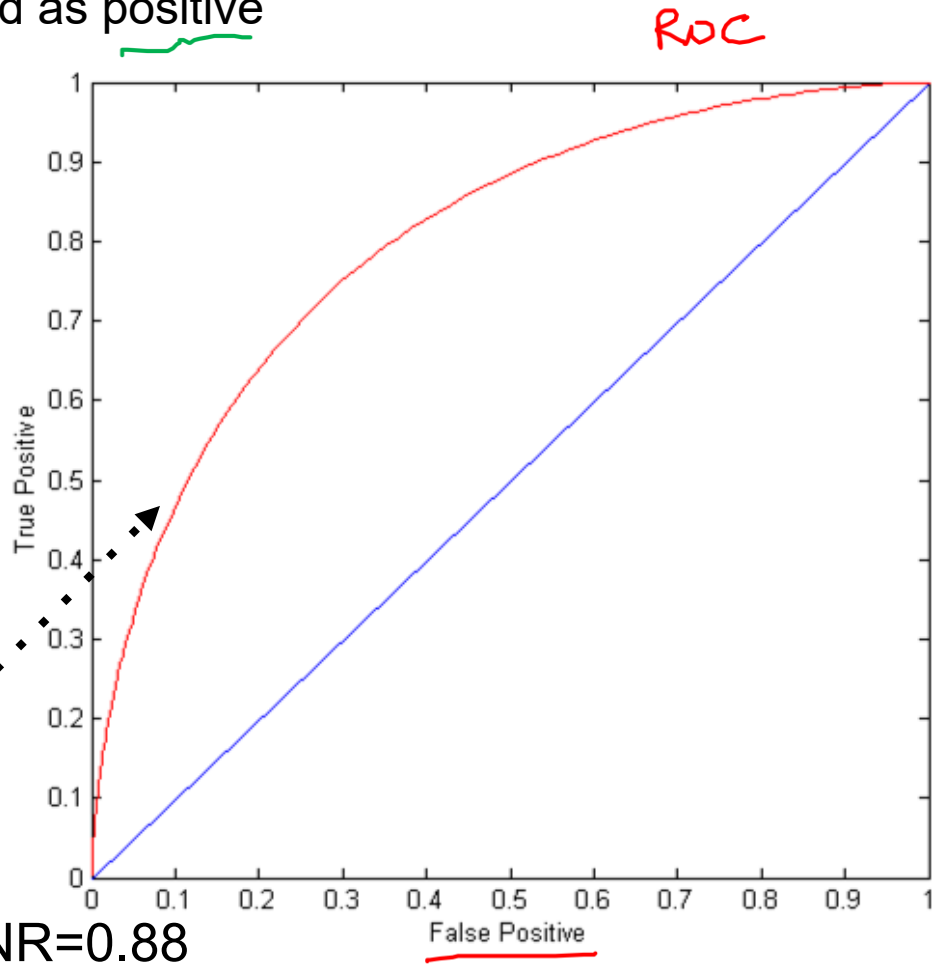
threshold	Accuracy
0.5	60%
0.6	70%
0.75	80%


Perfect classifier
ROC curve
1.0
Better
Worse
Random classifier
True positive rate
0.5
0.0
0.0
0.5
1.0
False positive rate

ROC Curve Example

- 1-dimensional data set containing 2 classes (positive and negative)
- Any points located at x > t is classified as positive





At threshold t:

TPR=0.5, FNR=0.5, FPR=0.12, TNR=0.88

How to Construct an ROC curve

Instance	Score	True Class
1	0.95	+
2	0.93	+
3	0.87	-
4	0.85	-
5	0.85	-
6	0.85	+
7	0.76	-
8	0.53	+
9	0.43	-
10	0.25	+

- Use a classifier that produces a continuous-valued score for each instance
  - The more likely it is for the instance to be in the + class, the higher the score
- Sort the instances in decreasing order according to the score
- Apply a threshold at each unique value of the score
- Count the number of TP, FP, TN, FN at each threshold
  - TPR = TP/(TP+FN)
  - FPR = FP/(FP + TN)

At Threshold 0.8

Any value ≥ 0.8 → positive / class 1

< 0.8 → class 0

Actual label	Predicted Value	Predicted label	
1	0.9	1	TP
0	0.8	1	FP
1	0.7	0	FN
0 ✓	0.4	0 ✓	TN
1	0.3	0	FN

TP=1, FP=1, FN=2, TN=1

TPR, FPR

How to construct an ROC curve

Class	+	-	+	-	-	-	+	-	+	+	
Threshold >=	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
TP	5	4	4	3	3	3	3	2	2	1	0
FP	5	5	4	4	3	2	1	1	0	0	0
TN	0	0	1	1	2	3	4	4	5	5	5
FN	0	1	1	2	2	2	2	3	3	4	5
TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0

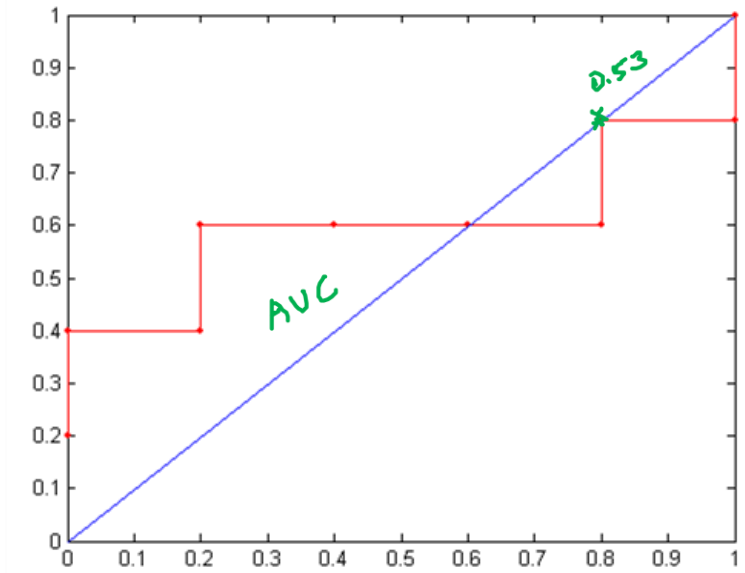
11 Thresholds

False Positive → labelled as positive incorrectly

Threshold ≥ 0.53 '+' class

< 0.53 '−' class

ROC Curve:



Instance	Score	True Class
1	0.95	+
2	0.93	+
3	0.87	-
4	0.85	-
5	0.85	-
6	0.85	+
7	0.76	-
8	0.53	+
9	0.43	-
10	0.25	+

Predicted label

+	TP
+	TP
+	FP
+	FP
+	FP
+	TP
+	FP
+	TP
−	TN
−	FN

To find the **best threshold**, we calculate **Youden's J**:

$$J = \text{TPR} - \text{FPR}$$

Example:

threshold

Instance	Score	True Class
1	0.95	+
2	0.93	+
3	0.87	-
4	0.85	-
5	0.85	-
6	0.85	+
7	0.76	-
8	0.53	+
9	0.43	-
10	0.25	+

lower threshold → more → aggressive False Positives

Recall ↑ Precision ↓

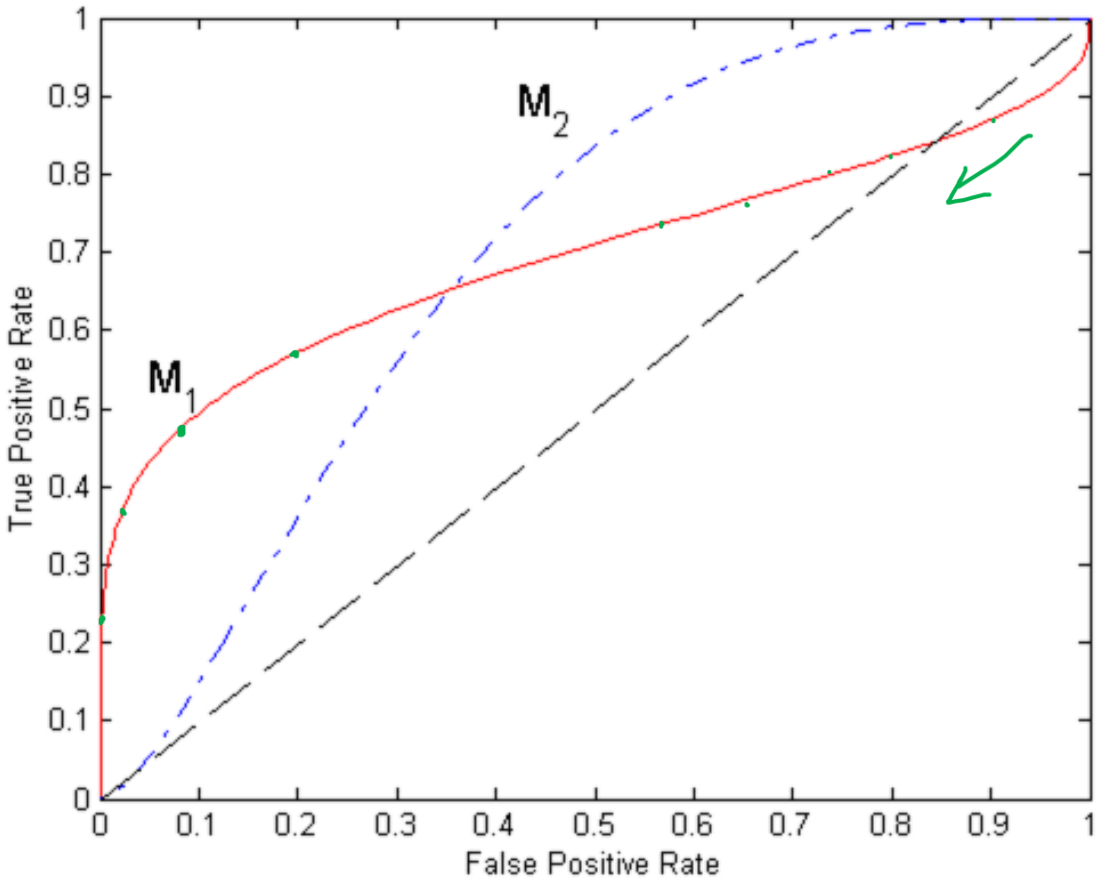
Higher threshold → Some positives will be missed
but the model is more conservative
Predicts only very confident positives

Recall ↓ Precision ↑

different thresholds Balance b/w Precision & Recall

1. Adjust the sensitivity (TPR) of our model
2. visualize performance across the range of operating pts (ROC, PR curve)

Using ROC for Model Comparison



- No model consistently outperforms the other
  - M_1 is better for small FPR
  - M_2 is better for large FPR
- Area Under the ROC curve (AUC)
  - Ideal:
    - Area = 1
  - Random guess:
    - Area = 0.5

AUC calculation

Using your ROC points (in decreasing FPR order):

(1.0, 0.8) → (0.8, 0.8) → (0.8, 0.6) → (0.6, 0.6) → (0.4, 0.6) → (0.2, 0.6) → (0.2, 0.4) → (0.0, 0.4) → (0.0, 0.2) → (0.0, 0.0)

Apply the trapezoidal rule $\mathbf{AUC} = \sum(FPR_i - FPR_{i+1}) \cdot \frac{TPR_i + TPR_{i+1}}{2}$:

- 1.0→0.8: width 0.2, avg TPR (0.8+0.8)/2=0.8 → area 0.16
- 0.8→0.6: width 0.2, avg TPR 0.6 → area 0.12
- 0.6→0.4: width 0.2, avg TPR 0.6 → area 0.12
- 0.4→0.2: width 0.2, avg TPR 0.6 → area 0.12
- 0.2→0.0: width 0.2, avg TPR 0.4 → area 0.08

(Vertical steps where width=0 add 0 area.)

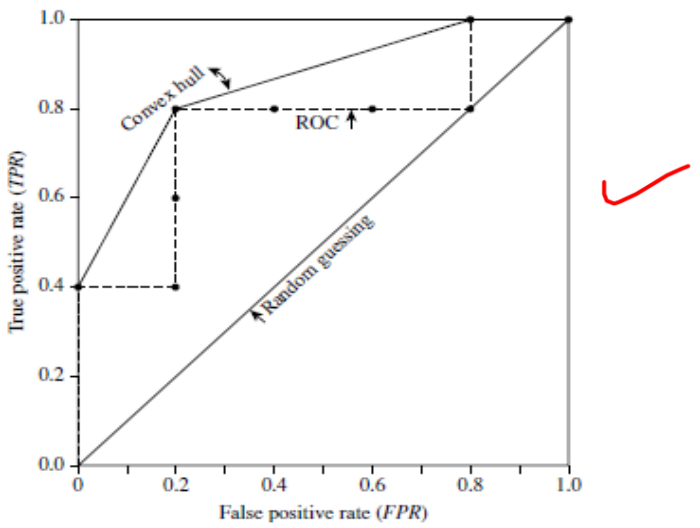
AUC = 0.16 + 0.12 + 0.12 + 0.12 + 0.08 = 0.60.

	FPR	TPR
i	1	0.8
i+1	0.8	0.8
i+2	0.8	0.6
	0.6	0.6
	0.4	0.6

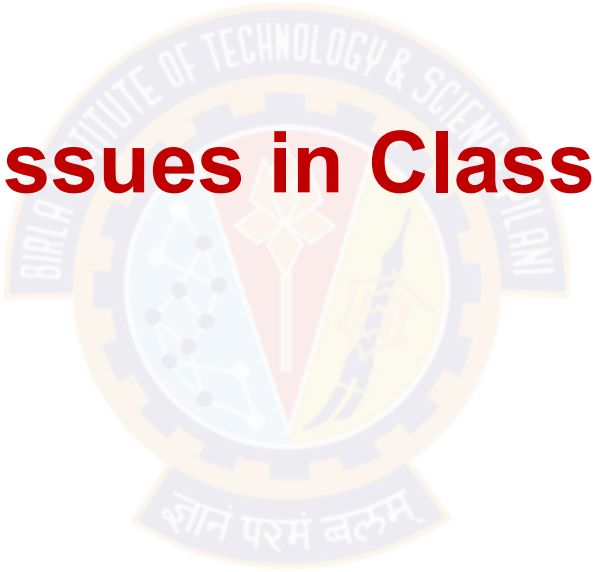
Example

The table below shows the probability value (column 3) returned by a probabilistic classifier for each of the 10 tuples in a test set, sorted by decreasing probability order. The corresponding ROC is given on right hand side.

Tuple #	Class	Prob.	TP	FP	TN	FN	TPR	FPR
1	*P*	0.90	1	0	5	4	0.2	0
2	*P*	0.80	2	0	5	3	0.4	0
3	*N*	0.70	2	1	4	3	0.4	0.2
4	*P*	0.60	3	1	4	2	0.6	0.2
5	*P*	0.55	4	1	4	1	0.8	0.2
6	*N*	0.54	4	2	3	1	0.8	0.4
7	*N*	0.53	4	3	2	1	0.8	0.6
8	*N*	0.51	4	4	1	1	0.8	0.8
9	*P*	0.50	5	4	0	1	1.0	0.8
10	*N*	0.40	5	5	0	0	1.0	1.0



Common Issues in Classifiers



Sigmoid Function

- The main class of interest is **rare**.
- The data set distribution reflects a significant majority of the negative class (Eg., Job Offered = Yes/1) and a minority positive class (Eg., Job Offered = No/0)

- For Another Example,
  - fraud detection applications, the class of interest (or positive class) is *"fraud,"*
  - *m*edical tests, there may be a rare class, such as *"cancer"*

- **Accuracy might not be a good option** for measuring performance in case of class imbalance problem

Solution to Class Imbalance

- Generate Synthetic Samples
- New samples based on the distances between the point and its nearest neighbors E.g. Synthetic Minority Oversampling Technique, or SMOTE class in sklearn
- Change the performance metric : Use Recall, Precision or ROC curves instead of accuracy
- Try different algorithms : Some algorithms as Support Vector Machines and Tree-Based algorithms may work better with imbalanced classes. We will discuss these post mid term

Logistic Regression –Additional Practice Exercises

CGPA	IQ	IQ	Job Offered
5.5	6.7	100	1
5	7	105	0
8	6	90	1
9	7	105	1
6	8	120	0
7.5	7.3	110	0

Hyper parameters:
Learning Rate = 0.8
Initial Weights = (-0.1, 0.2,-0.5)
Regularization Constant = 10

For this similar problem discussed in class note that the hyper parameters are different

1. Formulate the gradient descent update equations for this problem
2. Repeat the GD for two iterations
3. Find the Loss at every iterations and interpret your observation
4. Using the results of second iteration answer below questions:
   a) Interpret the influence of the CGPA in the response variable
   b) Predict if a new candidates with IQ=5 and CGPA = 9 will be offered job or not?
5. Repeat the steps 2 to 4 by using stochastic gradient descent instead of batch gradient descent for 4 iterations. (Take any random sample from among 6 instances for these 4 iterations)

Solution

For logistic regression with regularization:

$$h_\theta(x) = \frac{1}{1 + e^{-(\theta^T x)}}$$

Loss (with regularization):

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right] + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

Gradient Descent Update:

$$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)} + \frac{\lambda}{m} \theta_j \right]$$

where

- $\alpha = 0.8$ (learning rate)
- Initial weights = $[-0.1, 0.2, -0.5]$
- $\lambda = 10$ (regularization constant)

- For bias:

$$\theta_0 \leftarrow \theta_0 - 0.8 \cdot \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$$

- For $j = 1, 2$:

$$\theta_j \leftarrow \theta_j - 0.8 \left[\frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)} + \frac{10}{m} \theta_j \right]$$

Evaluation of Classifiers–Additional Practice Exercises

Given below is a confusion matrix for medical data where the class values are yes and no for a class label attribute, cancer. Answer the following questions.

Classes	*yes*	*no*	*Total*	*Recognition (%)*
yes	**90**	**210**	300	30.00
no	**140**	**9560**	9700	98.56
Total	230	9770	10,000	96.40

Confusion matrix for the classes *cancer* = *yes* and *cancer* = *no*.

1. Calculate the Precision , Recall, F-Score, Error-rate, ~~F-Score~~
2. Brainstorm on the use case / scenarios w.r.t given example, where precision is preferred over recall.
3. Brainstorm on the use case / scenarios w.r.t given example, where recall is preferred over precision.

Solution

Classes	yes	no	Total	Recognition (%)
yes	**90**	**210**	300	30.00
no	**140**	**9560**	9700	98.56
Total	230	9770	10,000	96.40

Confusion matrix for the classes *cancer* = *yes* and *cancer* = *no*.

Confusion matrix (given)

- **TP = 90** (pred yes, actual yes)
- **FN = 210** (pred no, actual yes)
- **FP = 140** (pred yes, actual no)
- **TN = 9560** (pred no, actual no)
- **Total = 10,000**

- **Precision (positive predictive value)**

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{90}{90 + 140} = \frac{90}{230} = 0.3913 \rightarrow \textbf{39.13\%}$$

- **Recall (sensitivity, true positive rate)**

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{90}{90 + 210} = \frac{90}{300} = 0.30 \rightarrow \textbf{30.00\%}$$

- **F1-score (harmonic mean of precision & recall)**

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$
$$= \frac{2 \cdot 0.3913 \cdot 0.30}{0.3913 + 0.30} \approx 0.3395 \rightarrow \textbf{33.95\%}$$

- **Specificity (true negative rate)**

$$\text{Specificity} = \frac{TN}{TN + FP} = \frac{9560}{9560 + 140} = \frac{9560}{9700} \approx 0.9856 \rightarrow \textbf{98.56\%}$$

- **False positive rate (FPR)** = $1 - \text{Specificity} = \frac{FP}{FP + TN} = \frac{140}{9700} \approx 0.0144 \rightarrow \textbf{1.44\%}$
- **False negative rate (FNR)** = $1 - \text{Recall} = \frac{FN}{TP + FN} = \frac{210}{300} = 0.70 \rightarrow \textbf{70.0\%}$
- **Negative predictive value (NPV)**

$$\text{NPV} = \frac{TN}{TN + FN} = \frac{9560}{9560 + 210} = \frac{9560}{9770} \approx 0.9785 \rightarrow \textbf{97.85\%}$$

- **Accuracy / Error rate**

$$\text{Accuracy} = \frac{TP + TN}{Total} = \frac{90 + 9560}{10000} = 0.965 \rightarrow \textbf{96.5\%}$$

Error rate = $1 - \text{Accuracy}$ = **3.5%** (350 errors / 10,000)

Solution

Prefer precision when **false positives are costly** (financially, medically, or ethically):

- A positive prediction triggers an **invasive, risky, or expensive** follow-up (e.g., unnecessary biopsy, surgery).
- Limited treatment/diagnostic resources — you want to ensure that those you call "positive" are likely true positives.
- Legal/operational contexts where a false alarm has large negative consequences (e.g., convicting somebody, mass evacuations).

Prefer recall when **missing a positive is very costly** (safety/health-critical):

- Disease screening where early detection substantially improves outcome (e.g., cancer screening). Missing cases (false negatives) is worse than having false alarms.
- Contagious disease control where undetected positives can infect many others.
- Any situation where follow-up tests are safe/cheap and you can filter false positives later.

In this cancer example: generally you want **high recall** for initial screening — catch as many cancers as possible, then confirm with more specific tests.

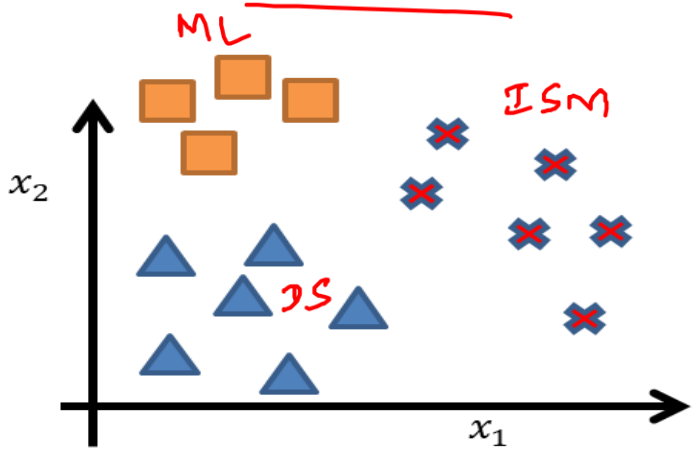
Types of Classification Based on the Output Labels

Types of Classification

- Target Concept

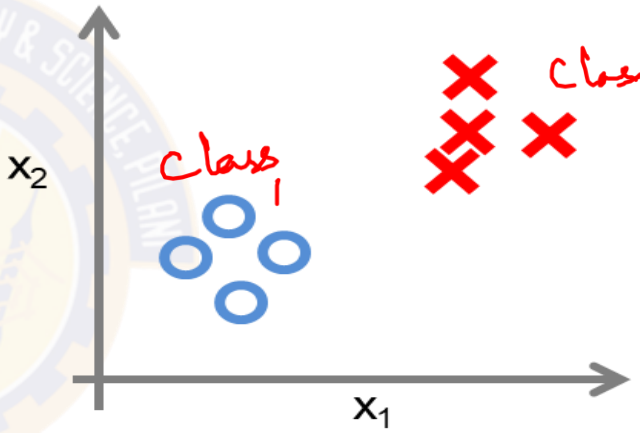
Output Labels

Multi Class ✓

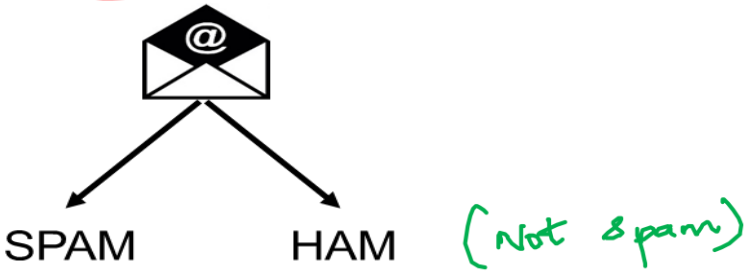


Enjoy Sports

Binary ✓



Spam Classifier ✓

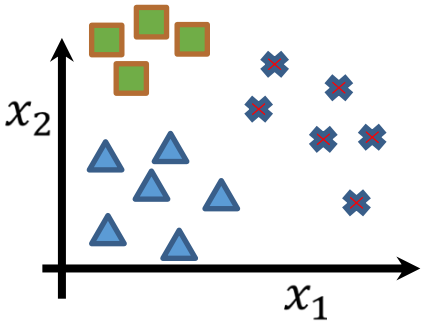


Prediction – Multi class Classification

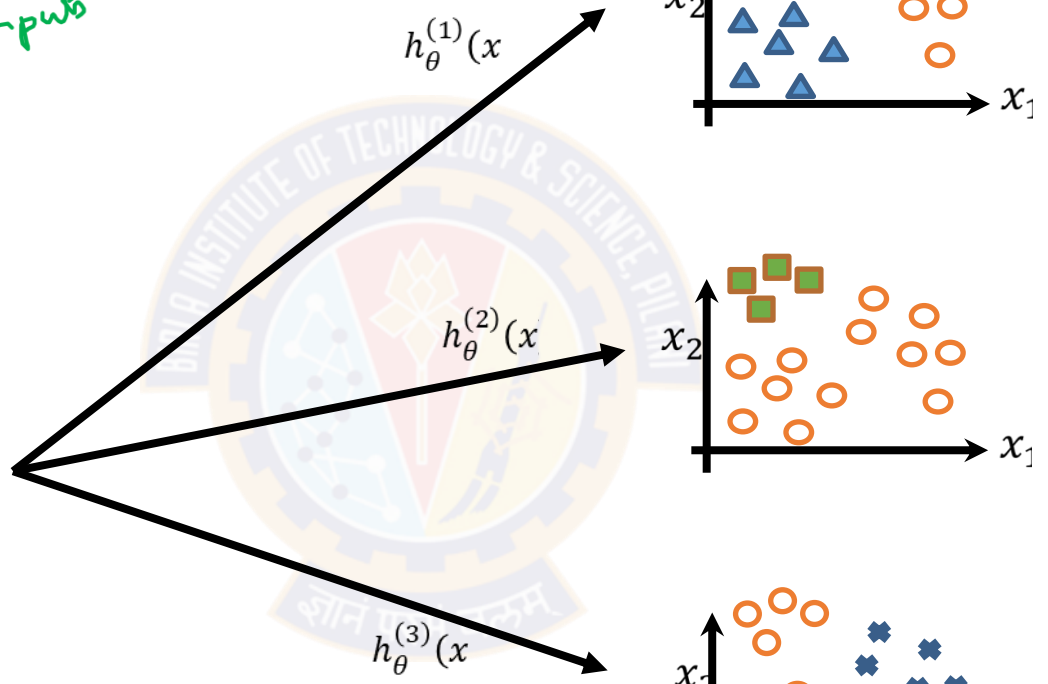
One Vs All Strategy(one-vs-rest)

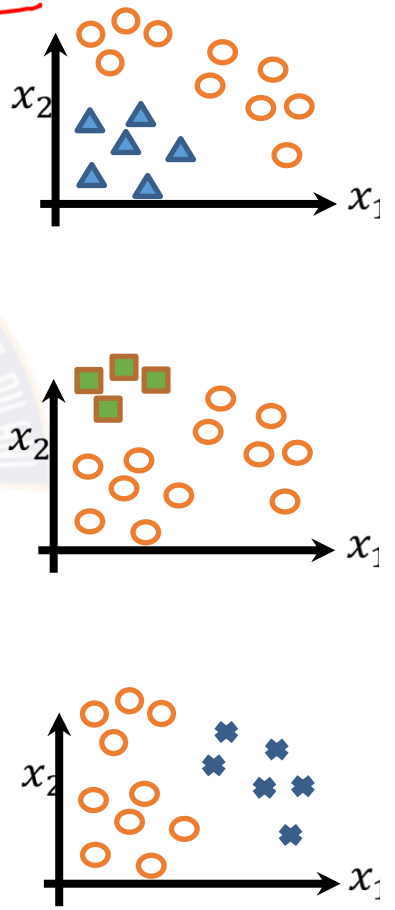
Multi-class inputs → 2-class o/p

Only two class labels



Class 1:
Class 2:
Class 3:



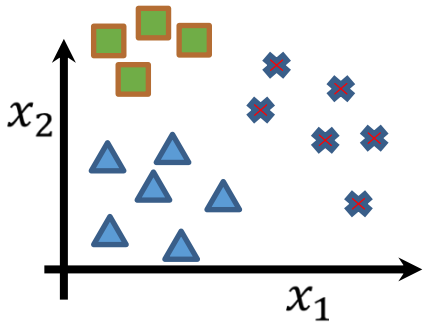


$$h_\theta^{(i)}(x) = P(y = i|x;\theta) \quad (i = 1,2,3)$$

$$\text{For input x Predict}: \max_i h_\theta^{(i)}(x)$$

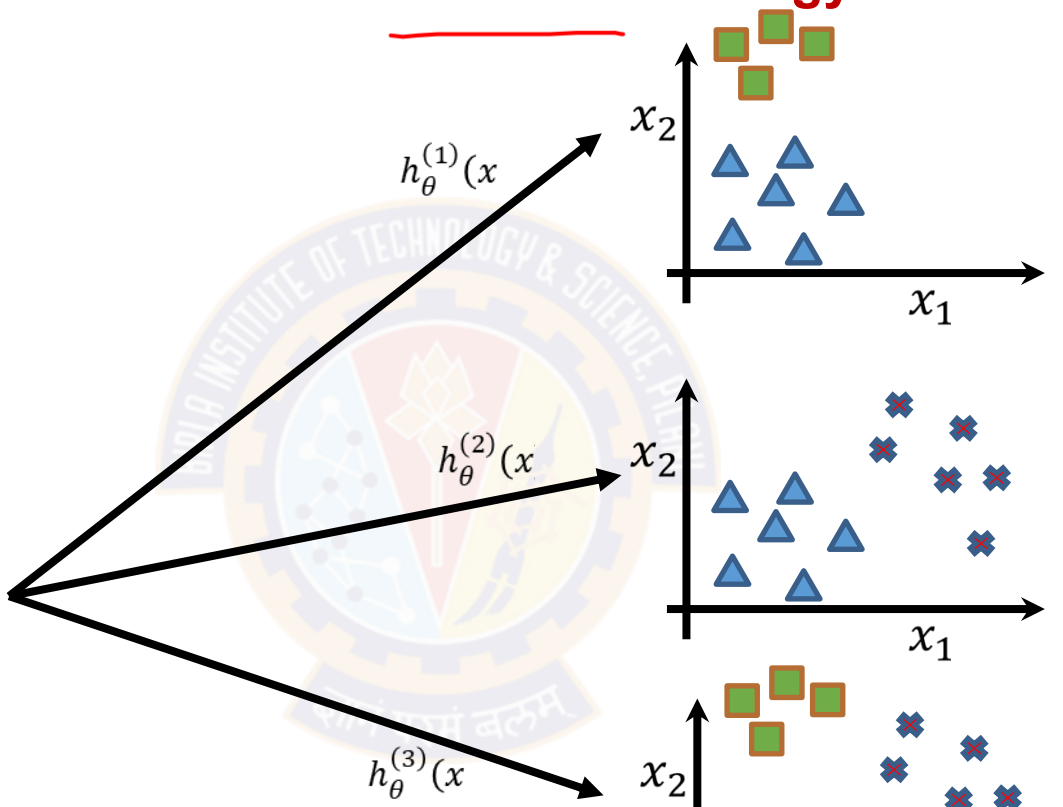
Note: Scikit-Learn detects when you try to use a binary classification algorithm for a multi-class classification task, and it automatically runs OvA (except for SVM classifiers for which it uses OvO)

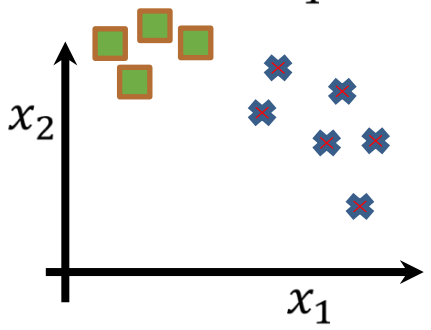
Prediction – Multi class Classification



Class 1:
Class 2:
Class 3:

One Vs One Strategy





$$h_\theta^{(i)}(x) = P(y = i|x;\theta) \quad (i = 1,2,3)$$

$$\text{For input x Predict}: \max_i h_\theta^{(i)}(x)$$

N × (N – 1) / 2 classifiers

Logistic regression (Classification)- Summary

- **Model**

$$h_\theta(x) = P(Y = 1|X_1, X_2, \cdots, X_n) = \frac{1}{1+e^{-\theta^\top x}}$$

- **Cost function**

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)}) \qquad \text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Odds = $\frac{p}{1-p}$

- **Learning**

Gradient descent: Repeat $\{\theta_j := \theta_j - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})\, x_j^{(i)}\}$

$p \rightarrow$ probability belonging to positive class

$1-p \rightarrow$ prob. belonging to negative class

- **Inference**

$$\hat{Y} = h_\theta(x^{\text{test}}) = \frac{1}{1 + e^{-\theta^\top x^{\text{test}}}}$$

Note:

- σ(t) < 0.5 when t < 0, and σ(t) ≥ 0.5 when t ≥ 0, so a Logistic model predicts 1 if xTθ is positive, and 0 if it is negative
- logit(p) = log(p / (1 - p)), is the inverse of the logistic function. Indeed, if you compute the logit of the estimated probability p, you will find that the result is t. The logit is also called the log-odds

Thank you !

Required Reading for completed session :

T1 - Chapter # 6 (Tom M. Mitchell, Machine Learning)

R1 – Chapter # 3,#4 (Christopher M. Bhisop, Pattern Recognition & Machine Learning) & Refresh your MFDS course basics

Next Session Plan :

Module 5 – Decision Tree Classifier